Crypto-Book: An Architecture for Privacy Preserving Online Identities

John Maheswaran, David Isaac Wolinsky, Bryan Ford
Yale University,
{john.maheswaran,david.wolinsky,bryan.ford}@yale.edu

ABSTRACT

Through cross-site authentication schemes such as OAuth and OpenID, users increasingly rely on popular social networking sites for their digital identities–but use of these identities brings privacy and tracking risks. We propose Crypto-Book, an extension to existing digital identity infrastructures that offers privacy-preserving, digital identities through the use of public key cryptography and ring signatures. Crypto-Book builds a privacy-preserving cryptographic layer atop existing social network identities, via third-party *key servers* that convert social network identities into public/private keypairs on demand. Using linkable ring signatures, these keypairs along with the public keys of other identities create unique pseudonyms untraceable back to the owner yet can resist anonymous abuse.

Our proof-of-concept implementation of Crypto-Book creates public/private key pairs for Facebook users, and includes a private key pickup protocol based on E-mail. We present Black Box, a case study application that uses Crypto-Book for accountable anonymous whistle-blowing. Black Box allows users to sign files deniably using ring signatures, using a list of arbitrary Facebook users – who need not consent or even be aware of this use – as an explicit anonymity set.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General — Security and protection; H.3.5 [**Information Storage and Retrieval**]: Online Information Services — Web-based services

General Terms

Design, Security

Keywords

Anonymity, anonymous communication, authentication, online social networks

1. INTRODUCTION

Social networks have gained widespread popularity among users as a means to managing their online identity; however, growing concern exists regarding online privacy. The extensive deployment of cross site authentication protocols such as OAuth [12] and OpenID [24] have simplified the processing of managing online identity across multiple sites. The unfortunate side effect, enables a user's online activity to be tracked more easily presenting a greater risk to online privacy.

We propose to leverage popular existing social network identities such as Facebook, but augment them with an additional cryptographic layer that enables more privacy-preserving uses of these existing identities. We distribute this privacy-preserving layer through one or more *key servers*, which could in principle be run either by the social networking sites themselves or by independent third-party organizations. These key servers convert social network identities into public/private keypairs. Users can then employ these keypairs in privacy-preserving, anonymous cryptographic protocols, through the use of ring signatures [26, 18] for example. We demonstrate these capabilities via Black Box, a case study for a deniable whistleblowing application.

We use an *anytrust* [33] (see Section 3.1) cloud of key servers that collaboratively assign key pairs to each social network identity. Users need only assume that *at least one* of these key servers is trustworthy, but need not know or care *which* one. All the other servers may be compromised without breaking the user's privacy protection. Users obtain their private key by authenticating with each of the key servers via OAuth. Each key server generates only one *part* of the user's private key; the user's client machine then combines these parts to obtain his full private key.

Any user can on demand request not only his own but any set of *other* users' *public* keys from these key servers. The public keys can then be used in arbitrary privacy-preserving cryptographic protocols that rely on public/private keypairs. In Crypto-Book, for example, a user can request several others' public keys from the keyservers, then use these public keys together with his own private key to create an anonymous *ring signature* [26, 18]. A ring signature proves that *one* of the "ring members," or one identities associated with the key set, signed the document but reveals no information about which one. Crypto-Book employs these ring signatures to authenticate anonymously or pseudonymously to third-party websites and services, without revealing personally identifying information, protecting users from being tracked across different web sites or services.

2. THREAT MODEL

We make the following assumptions about a potential adversary in the context of our system.

- We assume that the client has the ability to connect to the key servers through an anonymity network such as Tor [9].
- At least one of the key servers is honest, executing the protocols correctly and not sharing its master key or private keys with anyone else. This assumption follows the *anytrust* model [33].
- All key servers provide consistent public and private keys, and do not return two different results for two different requests for the same key.
- Dishonest servers may collude with each other to share their master secrets or private keys.
- Key servers can see the IP addresses of clients that connect to them.
- If an adversary compromises a server, they have access to the master key and all private keys from that server from that epoch but not from previous epochs, as detailed later in Section 3.4.

3. ARCHITECTURE

Figure 1 shows the overall system architecture. The client user logs into an online social networking site such as Facebook, and is provided with an OAuth token by the Facebook (or other social networking site) API.

The client then connects to each of the key servers in the anytrust cloud through an anonymity network such as Tor [9]. The connection between the client and each key server is secured using and end-to-end encrypted connection such as SSL. The client sends their OAuth token to each of the key servers. Each keyserver is tasked with maintaining a public/private keypair associated with each social network identity. These keypairs must be of a form such that they may be combined to make a composite public/private keypair, as is the case with DSA [10] keypairs. On receipt of the client's Facebook OAuth token, each keyserver obtains one part of the client's private key, and returns it to the client over the secure connection. See Section 3.3 for details on how these keys are obtained. Once the client has received responses from each of the key servers, the client

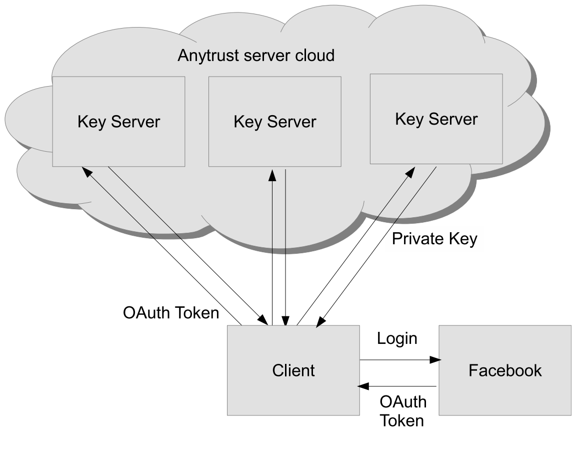

Figure 1: Overall System Architecture

combines the private keys to form a composite private key, which is the that user's overall private key.

Public keys are obtained similarly, except that it is not necessary to supply an OAuth token to the key servers when requesting a public key. The client simply specifies the Facebook username, or other social network identity information, and obtains public key parts from each of the key servers. The client then combines these parts to obtain the composite public key for the requested social network identity.

Once a client has obtained his own private key and a list of public keys corresponding to other Facebook identities, the client constructs a ring signature [26, 18], with all the Facebook identities acting as an explicit anonymity set. A ring signature has the property that a third party can verify, using only the public keys, that the signature was created by *some member* of this anonymity set. No one can determine *which* person in this anonymity set created the signature, however. The user can thus "conscript" arbitrary other social network users into an arbitrary ad hoc anonymity set, and "hide in" that set. If the user forms this ring from a meaningful group of other users – the board members of a company, for example – the user can sign documents or authenticate as an anonymous member of that designated group, perhaps conveying some degree of reputability or authority despite the user's anonymity.

The user may now use such ring signatures as a privacy-preserving form of online identity, in a multitude of possible scenarios. For example, the user could anonymously sign a document to give a credible leak, join an anonymous chat group open only to a specific set of users, or anonymously comment on blog posts.

3.1 Anytrust Server Model

Crypto-Book relies on a decentralized client/server model we dub *anytrust* [33]. In this model, each of many clients trust only that *at least one* of a smaller but administratively diverse set of servers behaves honestly. All but that one server might in principle behave arbitrarily maliciously and collude against the clients. Clients need not know which server to trust, but only trust that at least one is honest.

Under our threat model (Section 2) we assume at least one key server is honest, and other key servers may be colluding to compromise a client's private key. For each client, each key server i generates a private key part k_i. The client uses a combining function f that takes the private key parts $k_0, k_1, \ldots, k_n$ and combines them into a composite private key $k_c = f(k_0, k_1, \ldots, k_n)$. We choose the combining function f such that all n key parts k_i are required to calculate k_c, and given any $n-1$ key parts, it is cryptographically infeasible to calculate or learn any significant information about k_c.

3.2 Distributed Key Pickup Mechanism

The key pickup mechanism enables clients to pick up their private keys securely without an adversary being able to compromise their private keys. Figure 1 shows the key pickup protocol. To protect clients from being tracked by malicious servers, we assume clients connect to the key servers through an anonymity network such as Tor [9]. Through this anonymity network, the client opens an SSL connection to each of the keyservers. Beforehand the client obtains an OAuth token from its social networking provider, which the client sends over the SSL connection to each of the key servers.

On receipt of a private key request and OAuth token, the key server connects to the social networking provider using the OAuth token, to verify the user's identity and obtain the user's unique social networking username. If the user is successfully verified, the key server generates the private key for that user (as described in Section 3.3), and returns the private key part to the client over the SSL connection.

On receipt of all private key parts the client then combines these parts together into a composite key using a combining function as described in Section 3.1.

An alternative to having the client run software to collect the keys would be to have an intermediary server acting as a trusted web proxy whose job is is to authenticate the user, then collect the private key parts on the client's behalf, assemble them into the private key, and securely return this to the client. This approach eliminates the need for special client-side software, at the cost of requiring the client to trust the web proxy.

3.3 Key Generation by the Key Servers

Each of the key servers is tasked with generating a public/private keypair for each social networking identity. One way for the key servers to generate these keys is to generate a fresh random public/private keypair on each request for an unknown social network identity, then store the keypair for future reference.

Another approach is for a key server to hold a *master secret* or *master key*, from which the server creates or re-creates each client's private key deterministically on demand to serve a given request. For example, using a keyed pseudo-random number generator (PRNG), such as a hash message authentication code (HMAC), keyed on the master secret, the key server hashes the user's Facebook username to obtain the user's private key. This approach requires less storage, as the server need not store an ever-increasing number of client keys.

Another consideration is the type of keys generated by the servers. DSA [10] keys support easy key splitting and combining, enabling an anytrust group of servers, whereas RSA [27] keys do not easily support key splitting. We therefore prefer key schemes such as DSA that support key splitting.

3.4 Compromised Key Servers and Epochs

One risk is that a key server's storage might be compromised, for example if the master key is leaked or if a thief physically compromises the server. The adversary would obtain all the private keys saved on disk, or the master secret from which all private keys may be generated, either way compromising all users' private keys. While one server's compromise does not reveal the user's composite private key if the server is only of an anytrust group, we wish to limit the damage even if *all* servers in a group are *eventually* broken. We propose an epoch based scheme to limit such damage.

We divide the key server's work into epochs, where the keyserver's master secret is valid only during a given epoch, and gets randomly reinitialized in each successive epoch. If we want previously generated ring signatures to be verifiable after the epoch, the server must maintain a list of public keys containing the public keys generated in that epoch. In subsequent epochs the server can serve requests for older public keys, but not private keys, to allow verification of old ring signatures.

Since the master secret gets randomly reinitialized in each successive epoch, each user can thus get a new public/private keypair in each key server epoch. The key server can create (or recreate) a client's private key only for the current epoch, limiting the damage of a server compromise to the current epoch.

The epoch based key scheme used in conjunction with the anytrust key splitting scheme significantly reduces the risk of a client's private key being compromised by an adversary. As long as at least one honest server securely erases its master secret at the end of each epoch, private keys generated during that epoch are not compromised even if all servers are later compromised.

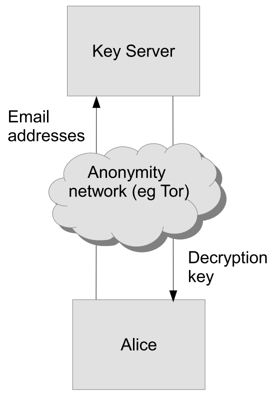

Figure 2: Anonymous private key request

3.5 Anonymous Key Pickup

One threat arises from *intersection attacks* [25, 15, 6] where a compromised key server allows an adversary to see who requests their private keys. When subsequent ring signatures are created and used, the adversary might deduce who created a ring signature, for example if only a small subset of the anonymity set had collected their private keys. We propose an *anonymous key pickup* protocol that helps to protect client anonymity in the face of such intersection attacks.

Suppose Alice wishes to collect her private key part from a key server, but we do not want the key server to know that Alice has picked up her private key. Alice connects through an anonymity network to the key server using a secure SSL connection, but this time does not immediately log in with Facebook, or otherwise identify herself to the key server. Alice instead merely supplies the key server a list of email addresses representing a desired anonymity set, including her own email address, as shown in Figure 2. The server generates a private key for each email address, and encrypts each private key using the same single symmetric key. The server sends over the secure connection to Alice the symmetric encryption key along with instructions on how to decrypt her private key.

Each encrypted private key is then attached to an email inviting that user to sign up to the service, as shown in Figure 3. The emails are then sent such that each email address only receives its own encrypted private key. Alice checks her email, finds the attachment to the invitation, and decrypts it to obtain her private key part. To prevent the system from sending out too many messages rate limiting could be employed.

Since her private key was encrypted, her email provider or anyone who compromises her email or intercepts does not have access to it. Additionally, since the server received an anonymous request and sent out multiple private keys to multiple emails, the server does not know who in the end was able to decrypt and use their private

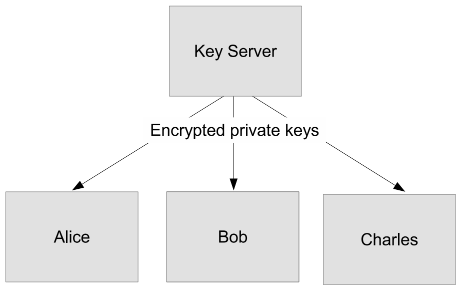

Figure 3: Invitation emails with encrypted private keys. Alice can decrypt her private key.

key, hence protecting Alice against intersection attacks from a compromised server.

Alice needs to carry out this scheme only once per key server: e.g., if with three key servers, Alice need only do it three times. Alice can then construct her private key and keep it saved for future use so as to avoid having to participate in anonymous key distribution again.

In addition to email, this protocol could also be deployed over other communication channels such as social network messaging or SMS text messaging.

3.6 Ring signatures

Ring signatures [26] build on group signatures [3], in that a message signed with a ring signature is endorsed by *some* member of a particular group, without revealing *which* member's private key was used to produce the signature. Ring signatures can be created on an ad hoc basis using arbitrary sets of public keys. The owners of those public keys need not "consent" or even be aware that they are being "conscripted" into an anonymity set for signing purposes.

Ring signatures are used in our architecture to provide an anonymity preserving identity to a user. The ring signature may be used to authenticate with a third party website or service so the third party knows that the user is a member of a group of users, but the specific user's identity is not revealed to the third party.

Rivest *et al.*'s ring signatures assume all ring members have an RSA public/private keypair. The signer must obtain the public keys of all other parties to use in the signer's anonymity set, in addition to their own private key, to generate a ring signature. These ring signatures offer *forward anonymity*: even with access to all the private keys, given an RSA ring signature, an attacker cannot unmask the original signer. RSA keys do not readily support key splitting, however, thus requiring dependence on a single trusted key server.

Linkable ring signatures (LRS) [18] instead use DSA-style [10] keys and the discrete logarithm problem. Linkable ring signatures have the additional property of *linkability*: given any two signatures, a third party can determine whether they were produced by the same or different members of the ring. By using LRS to authenticate anonymously to third-party web services, for example, the web service can ensure that the owner of a given underlying social network identity obtains *one and only one* pseudonymous account on the service, reducing vulnerability to anonymous abuse or trolling.

Because they use DSA-style keys, LRSs support key splitting, and hence anytrust key server groups. LRSs do not provide *forward anonymity* as basic ring signatures do, however. If a private key is compromised, then the attacker might use this key to unmask previously generated LRSs and discover whether they were indeed produced by that private key. This weakness is an important issue to be explored in future work.

4. PROTOTYPE IMPLEMENTATION

To demonstrate and evaluate our privacy-preserving architecture we implemented *Crypto-Book*, a proof-of-concept prototype that allows a user to log in with Facebook, and collect a private key from a group of key servers. We also implemented *Black Box*, a case study application built on the Crypto-Book framework. Black Box allows a user to sign a file anonymously, for example if they wish to leak a document, while offering evidence of the document's credibility in the form of a proof that it comes from one of a relevant group of Facebook users. Crypto-Book is available online for experimentation[1], along with full source code[2].

We used Facebook as the social networking provider and connected using OAuth and the Facebook Graph API[3]. We implemented both DSA [10] and RSA [27] based key pickup systems. DSA supports key splitting whereas RSA does not. For the RSA version we implemented a single key server and for the DSA version we implemented a three server anytrust group with the keys split over the three servers (details in Section 4.1).

For Black Box we implemented a downloadable signing application that allows users to collect their private keys from the anytrust servers and use them to sign a file anonymously. This application requires an OAuth token, which the user can obtain by authenticating with Facebook through the Crypto-Book site.

To simplify deployment, we also make this application available on a server that acts as a trusted web proxy, where a user can collect their private key parts, assemble them into the composite key, and collect other users' public keys. The server also allows users to upload files, to be signed anonymously using a ring signature on the server. This functionality is also available in the downloadable application, if the user does not wish to trust the web proxy.

[1]`http://www.crypto-book.com`
[2]`https://github.com/jyale/blackbox`
[3]`https://developers.facebook.com/`

4.1 DSA Based Scheme

We implemented an LRS scheme based on DSA keys. DSA keys operate in a group G of order p and are of the form $Y = g^x \bmod p$, where Y is the public key and x is the private key. A composite key can be formed from a set of keys by adding the private keys and multiplying the public keys.

We implemented three-server distributed key pickup. We implemented *two alternative ways* of collecting and assembling the key parts into a composite key:

- A downloadable application allowing the user to pickup and assemble the key parts on their machine
- A trusted web proxy

Key distribution works as follows: the user logs into Facebook, either through the trusted web proxy, or through our site that provides them with an OAuth authentication token. If the user is using the desktop app, they then supply this token to the app. Once the user is authenticated, the OAuth authentication token is sent to each of the three keyservers to request the private key.

Once a keyserver receives a private key request along with an OAuth token, it makes a request to the Facebook API to verify the token and obtain the user's corresponding Facebook username. If a valid username is returned (the authentication succeeds) then the keyserver will look up the corresponding private key in its database and return it to the requester (the proxy or the desktop app). For public key requests the requester sends to the keyserver the Facebook username that they want to obtain the public key for, the keyserver will look up the key and return it to the requester. If for any request the server does not already have a keypair saved for that Facebook username, the server will generate a keypair and store it in its database, returning the appropriate key to the requester.

Once the requester receives responses from all the servers, it will compute the composite private and public keys. The requester now has the required list of public keys and the private key needed and generates the linkable ring signature for the specified file. The LRS and the file can now be published through an anonymity network such as Tor [9].

4.2 Anonymous email key pickup

We implemented the anonymous key pickup protocol described in Section 3.5 over email. The key servers maintain a list of public keypairs, one corresponding to each email address. The first time an email address is included in an anonymity set, a key pair is generated for that email address and stored on the servers.

A client makes a request to a key server (or all anytrust servers) by supplying a list of email addresses. The key server obtains the private keys corresponding to each email address. Each of these private keys are then themselves encrypted using a fresh symmetric key. Each encrypted key is then emailed out to the corresponding email address for that key, disguised as an invitation email to use the Crypto-Book Black Box service, with the encrypted key attached. The symmetric key required to open this encrypted private key is provided onscreen to the original requesting user. They use this symmetric key to decrypt the private key they received via their email account.

4.3 Future Work

One of the most interesting areas for future work may be in investigating the impact different group/anonymity set choices have on user privacy protection. A possible threat to user privacy comes from the fact that third party sites may collude to attempt to deanonymize and uniquely identify users using some form of cross-site correlation attack. If a user authenticates himself as a member of a group across many third party sites, this vector of group membership may threaten the user's anonymity. If different sites use different groups there may be some risks to user privacy, the extent of which would depend on the way groups are chosen.

Having Crypto-Book batch users into groups, which are used across all third party sites and services, might enhance privacy. It may be possible to improve usability by having different user groups defined for different third party services. For example, a user could be a member of a special interest group on one site and a geographical group on another site. This may improve the usability of the third party sites, but future work is required to investigate how this custom group definition can be carried out without threatening user privacy.

We plan to look at how our architecture can integrate with popular software such as Wikimedia, to provide anonymous identities that allow anonymous editing, while mitigating abuses such as sock-puppetry and vandalism. Another line of investigation would be to integrate our key pickup protocol with an anonymous group chat system, such as Dissent [4, 34]. We will also explore how our privacy protecting identities can be tied back into anonymous posting within Facebook, as proposed in Faceless [29].

A further line of investigation would be looking at the usability of our solution. A user study might be helpful to compare how user friendly our solution is in comparison with traditional cross-site Facebook authentication, or with other approaches to authentication. We also plan to look at the the practicality of our approach. In particular we could look at overheads such as signature generation and verification time, and signature size.

Integrating identity based encryption [1] would allow users to avoid having to request individual public keys, after obtaining the IBE parameters and the key server's master public key.

5. RELATED WORK

The deployment of public key cryptography over social networks was considered by Narayanan *et al.* [23], who explored using social networks as a public key infrastructure (PKI), but did not implement any applications that use such public keys.

Various schemes have been proposed to protect user data *within* an online social network [20, 19, 11, 5, 14], by encrypting the content stored within the social network. However these schemes did not consider the privacy risks involved when a user uses their online social networking identity to identify themselves with third parties such as logging into other websites using their Facebook credentials.

PseudoID [8] is a similar system based on blind signatures [2] for privacy protected federated login. This scheme does not handle key assignment or Sybil resistance, as our work does. A similar blind signature based system was proposed by Khattak *et al.* [16]. Watanabe and Miyake [32] made initial efforts towards account checking, but did not consider key assignment. Opaak [21] attempts to provide some Sybil resistance by relying on a cellphone as a scare resource. SudoWeb [17] looked at limiting the amount of Facebook information disclosed to third party sites, but did not consider fully anonymous online IDs.

In identity based encryption (IBE), a public key can be an arbitrary string, such as a user's email address or social security number. This idea was first proposed by Shamir [28] and since then several IBE systems have been proposed [1, 7, 13, 22, 31, 30]. Our key servers collectively act similarly to an IBE PKG. Our approach avoids the need for pairing based cryptography, at the cost of requiring a separate key server request to obtain a public key for each user.

6. CONCLUSIONS

Our architecture demonstrates a usable, anonymous way to provide online social network users with privacy preserving online identities. We believe there are a large number of areas for future research based on our architecture, as well as a multitude of applications that could be developed on top of our framework.

7. ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their constructive feedback and comments.

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) SAFER contract N66001-11-C-4018. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA.

8. REFERENCES

[1] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *21st CRYPTO*. 2001.

[2] D. Chaum. Blind signatures for untraceable payments. In *CRYPTO*, 1982.

[3] D. Chaum and E. V. Heyst. Group signatures. In *Eurocrypt*, Apr. 1991.

[4] H. Corrigan-Gibbs and B. Ford. Dissent: accountable anonymous group messaging. In *17th CCS*, Oct. 2010.

[5] L. A. Cutillo, R. Molva, and T. Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *Communications Magazine, IEEE*, 2009.

[6] G. Danezis and A. Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *Information Hiding Workshop*, May 2004.

[7] Y. Desmedt and J.-J. Quisquater. Public-key systems based on the difficulty of tampering (is there a difference between DES and RSA?). In *CRYPTO*, 1987.

[8] A. Dey and S. Weis. PseudoID: Enhancing privacy in federated login. *HotPETs*, 2010.

[9] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *12th USENIX Security*, Aug. 2004.

[10] Federal Information Processing Standards Publication. Digital signature standard (DSS), July 2013. FIPS 186-4.

[11] S. Guha, K. Tang, and P. Francis. Noyb: Privacy in online social networks. In *WOSN*, 2008.

[12] E. Hammer-Lahav. The OAuth 1.0 protocol, Apr. 2010. RFC 5849.

[13] D. Hühnlein, M. Jacobson, and D. Weber. Towards practical non-interactive public key cryptosystems using non-maximal imaginary quadratic orders. In *Journal Designs, Codes and Cryptography*, 2003.

[14] S. Jahid, P. Mittal, and N. Borisov. EASiER: Encryption-based access control in social networks with efficient revocation. In *ASIACCS*, 2011.

[15] D. Kedogan, D. Agrawal, and S. Penz. Limits of anonymity in open environments. In *5th International Workshop on Information Hiding*, Oct. 2002.

[16] Z. A. Khattak, J.-l. A. Manan, S. Sulaiman, et al. Analysis of open environment sign-in schemes-privacy enhanced & trustworthy approach. *Journal of Advances in Information Technology*, 2011.

[17] G. Kontaxis, M. Polychronakis, and E. P. Markatos. SudoWeb: Minimizing information disclosure to third parties in single sign-on platforms. In *ISC*. 2011.

[18] J. K. Liu, V. K. Wei, and D. S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In *Australian Conference on Information Security and Privacy*, pages 614–623, July 2004.

[19] M. M. Lucas and N. Borisov. flyByNight: mitigating the privacy risks of social networking. In *SOUPS*, 2009.

[20] W. Luo, Q. Xie, and U. Hengartner. FaceCloak: An architecture for user privacy on social networking sites. In *CSE*.

[21] G. Maganis, E. Shi, H. Chen, and D. Song. Opaak: using mobile phones to limit anonymous identities online. In *MobiSys*, 2012.

[22] U. Maurer and Y. Yacobi. Non-interactive public-key cryptography. In *EUROCRYPT*, 1991.

[23] A. Narayanan. SocialKeys: Transparent cryptography via key distribution over social networks. In *IAB Workshop on Internet Privacy*, 2010.

[24] OpenID. `http://openid.net/`.

[25] J.-F. Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In *Design Issues in Anonymity and Unobservability*, July 2000.

[26] R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *ASIACRYPT*, pages 552–565, Dec. 2001.

[27] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 1978.

[28] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in cryptology*, 1985.

[29] X. Song, D. I. Wolinsky, and B. Ford. Faceless: decentralized anonymous group messaging for online social networks. In *SNS*, April 2012.

[30] H. Tanaka. A realization scheme for the identity-based cryptosystem. In *CRYPTO*, 1987.

[31] S. Tsujii and T. Itoh. An ID-based cryptosystem based on the discrete logarithm problem. *IEEE Journal on Selected Areas in Communications*, 1989.

[32] R. Watanabe and Y. Miyake. Account management method with blind signature scheme. *Engineering and Technology, World of Science*, 2011.

[33] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson. Scalable anonymous group communication in the anytrust model. In *EuroSec*, Apr. 2012.

[34] D. I. Wolinsky, H. Corrigan-Gibbs, A. Johnson, and B. Ford. Dissent in numbers: Making strong anonymity scale. In *10th OSDI*, Oct. 2012.